var get\_name = document.getElementById("create\_name");

var get\_inversion = document.getElementById("create\_inversion");

var get\_inputs = document.getElementById("create\_inputs");

var get\_gateDelay = document.getElementById("create\_gateDelay");

var get\_logicStart = document.getElementById("create\_startLogic");

var get\_logicalHIGH = document.getElementById("logical\_H");

var get\_logicalLOW = document.getElementById("logical\_L");

var logic\_functionStorage = document.getElementById('function\_selection');

var select\_element = document.getElementById("edit\_IC");

var edit\_box = document.getElementById("editor\_container");

var logic\_label = 0;

function initialization()

{

logic\_label = 0;

}

function IC\_objectCreator()

{

var name\_taken = false;

//Checks if logic\_lines is empty and if line is not named

if (logic\_lines.length == 0 && get\_name.value.trim() == "")

{

name\_taken = true;

get\_name.style.backgroundColor = "hsla(204,100%,50%,0.5)";

get\_name.value = "";

get\_name.placeholder = "Please enter a name";

}

else

{

/\*For loop checks if the name of the line is already taken or there is no

name or is only whiteSpace. If any of these are true, then the user

will be notified.

\*/

for (var x = 0; x < logic\_lines.length; x++)

{

//Checks if the name is already taken

if (get\_name.value == logic\_lines[x].name)

{

name\_taken = true;

get\_name.style.backgroundColor = "hsla(0,100%,50%,0.5)";

get\_name.value = "";

get\_name.placeholder = "Name already taken";

break;

}

//Checks if fields is empty or has only whiteSpace

if (get\_name.value.trim() === "")

{

name\_taken = true;

get\_name.style.backgroundColor = "hsla(204,100%,50%,0.5)";

get\_name.value = "";

get\_name.placeholder = "Please enter a name";

break;

}

}

//If nothing is wrong, a new logic object is created

if (!name\_taken)

{

get\_name.style.backgroundColor = "white";

//eval() is used to create multiple variables with different names

eval('var IC'+

logic\_label+

' = new IC\_object();'

);

//The new logic object is added to logic\_lines array

logic\_lines.push(eval("IC"+logic\_label));

IC\_createFunction();

load\_names();

reset\_logicClock();

get\_logicStart.value = false;

//logic\_label counts up by one for next valid logic object

logic\_label++;

}

}

}

//IC\_object holds the parameters for the new logic object

function IC\_object()

{

this.name = get\_name.value;

this.location = (logic\_lines.length+1)\*(logic\_vary+logic\_space);

//Turns inverted input from a string to a boolean with eval()

this.inverted = eval(get\_inversion.value);

//Function prepares array

this.inputs = input\_orderer(false);

this.start = eval(get\_logicStart.value);

this.input\_iter = 0;

this.gateDelay = Number(get\_gateDelay.value);

this.gate\_delayLevel = 0;

this.prev\_logic = 0;

this.gate\_signalChange = 0;

this.gate\_increment = 0;

this.logic\_output = [0,0,0,

function()

{

return this[0]-this[1]\*this[2];

}];

this.logic\_change = [get\_logicalHIGH.value, get\_logicalLOW.value];

this.logic\_state = "LOW";

this.logic\_function = [0, logic\_functionStorage.value];

}

//Converts inputs from string to a numerically ordered array

function input\_orderer(edit\_flag)

{

if (edit\_flag)

{

var input\_array = edit\_inputs.value.split(" ");

}

else

{

var input\_array = get\_inputs.value.split(" ");

}

var ordered\_array = [];

//Checks if array contains letters or unnecessary zeros

for (var x = 0; x < input\_array.length; x++)

{

var number\_checker = Number(input\_array[x]);

if (!Number.isInteger(number\_checker) || input\_array[x] == 0)

{

continue;

}

ordered\_array.push(input\_array[x]);

}

ordered\_array = ordered\_array.sort(function(a, b){return a - b});

var pruned\_array = [];

/\*For loop below checks if any duplicate numbers are present by comparing their position

in the ordered\_array to the first time the number is seen. If the location does not

match, the number is skipped over, exluding it from pruned\_array.

\*/

for (var x = 0; x < ordered\_array.length; x++)

{

var search\_array = ordered\_array[x].toString();

if (ordered\_array.indexOf(search\_array) != x)

{

continue;

}

pruned\_array.push(ordered\_array[x]);

}

return pruned\_array;

}

function order\_existingArray(IC\_lineReorder)

{

var reorder\_line = logic\_lines[IC\_lineReorder];

var reorder\_array = reorder\_line.inputs.sort(function(a, b){return a - b});

var repruned\_array = [];

/\*For loop chacks if any value is repeated by comparing its position to

the value's first appearance in the array

\*/

for (var x = 0; x < reorder\_array.length; x++)

{

var search\_array = reorder\_array[x];

if (reorder\_array.indexOf(search\_array) != x)

{

continue;

}

repruned\_array.push(reorder\_array[x]);

}

reorder\_line.inputs = repruned\_array;

}

function IC\_createFunction()

{

if (logic\_lines[logic\_label].logic\_function[1] !== 'null')

{

logic\_lines[logic\_label].logic\_function[0] = 1;

}

logic\_functionStorage.value = 'null';

}

/\*In order to load all the line names properly using createElement object,

you need to leave the scope in which the create object is located.

So the function below is used to call and exit the scope of the

editor\_nameLoader.

\*/

function editor\_nameLoaderIter()

{

select\_element.innerHTML = ''; //Clears selection element

//for loop used to call the name loader function

for (var x = 0; x < logic\_lines.length; x++)

{

editor\_nameLoader(x);

}

}

function editor\_nameLoader(iter)

{

var option\_element = document.createElement("OPTION");

/\*The create option element needs to be local to function to create

multiple option selectors

Using the for loop iteration, the logic name and option value is loaded,

then the option is loaded into the editor select element.

\*/

option\_element.innerHTML = logic\_lines[iter].name;

option\_element.value = iter;

select\_element.appendChild(option\_element);

}

function IC\_editor()

{

var edit\_name = logic\_lines[select\_element.value].name;

//edit\_inptStr needs to be converted to a string first before string methods will work

var edit\_inputStr = logic\_lines[select\_element.value].inputs.toString();

var edit\_logicH = logic\_lines[select\_element.value].logic\_change[0];

var edit\_logicL = logic\_lines[select\_element.value].logic\_change[1];

var edit\_srtlogic = logic\_lines[select\_element.value].start;

var edit\_invert = logic\_lines[select\_element.value].inverted;

var edit\_gateDely = logic\_lines[select\_element.value].gateDelay;

edit\_box.innerHTML =

'Name:<input id="edit\_name" type="text" value=""><br>' +

'Inputs:<input id="edit\_inputs" type="text" value=""><br>' +

'Logical HIGH:<input id="edit\_logical\_H" type="number" value="">%<br>' +

'Logical LOW:<input id="edit\_logical\_L" type="number" value="">%<br>' +

'Logic Start:<select id="edit\_startLogic">' +

'<option value="false">LOW</option>' +

'<option value="true">HIGH</option>' +

'</select><br>' +

'Inversion:<select id="edit\_inversion" value="true">' +

'<option value="false">false</option>' +

'<option value="true">true</option>' +

'</select><br>' +

'Gate Delay:<input id="edit\_gateDelay" type="number" value=""><br>' +

'<button onclick="change\_ICValues(select\_element.value)">Save Changes</button>' +

'<button onclick="close\_editBox()">Cancel</button>';

document.getElementById("edit\_name").value =

edit\_name;

document.getElementById("edit\_inputs").value =

edit\_inputStr.replace(",", " ");

document.getElementById("edit\_logical\_H").value =

edit\_logicH;

document.getElementById("edit\_logical\_L").value =

edit\_logicL;

document.getElementById("edit\_startLogic").value =

edit\_srtlogic;

document.getElementById("edit\_inversion").value =

edit\_invert;

document.getElementById("edit\_gateDelay").value =

edit\_gateDely;

}

function change\_ICValues(edit\_ICLine)

{

var get\_editName = document.getElementById("edit\_name");

var get\_editInversion = document.getElementById("edit\_inversion");

var get\_editInputs = document.getElementById("edit\_inputs");

var get\_editGateDelay = document.getElementById("edit\_gateDelay");

var get\_editLogicStart = document.getElementById("edit\_startLogic");

var EDIT = logic\_lines[edit\_ICLine];

EDIT.name = get\_editName.value;

EDIT.inverted = eval(get\_editInversion.value);

EDIT.inputs = input\_orderer(true);

EDIT.start = eval(get\_editLogicStart.value);

EDIT.gateDelay = Number(get\_editGateDelay.value);

close\_editBox()

reset\_logicClock();

load\_names();

}

function close\_editBox()

{

edit\_box.innerHTML = "";

}

/\*

document.getElementById("test").innerHTML =

ordered\_array.indexOf(search\_array);

var new\_inputs = {

name:'CLK', //Name of logic line

location:15, //Location of logic line on canvas

inverted:false, //Weather or not the signal is inverted

/\*Logic inputs. INPUTS MUST BE IN AN ARRAY!! Sort method

\* numerically sorts the input values

inputs:test\_inputs.sort(function(a, b){return a - b}),

start: true, //Starting position of logic

input\_iter:0, //Used to keep cycle through the inputs

gateDelay:6, //How long the signal takes to change from HIGH to LOW

gate\_delayLevel:0, //Used to save logic location

prev\_logic:0, //Used to save previous logic state

gate\_signalChange:0, //Indicates if the logic goes LOW to HIGH of HIGH to LOW

gate\_increment: 0, //Used to calculate height of logic level with a gate delay

/\*key: [not, flip, change, function]

\* [draw\_logic] saves the first three variables to use later in

\* calculating the location of the previous and current logic.

logic\_output:[0,0,0,

function ()

{

return this[0]-this[1]\*this[2];

}]

this.logic\_change = [get\_logicalHIGH.value, get\_logicalLOW.value]; //Stores values for signal hysteresis

this.logic\_state = "LOW"; //Holds what logic state the line has

this.logic\_function = [0, logic\_functionStorage.value]; //Stores logic gate information

};

\*/